PDFからテキストを抽出する

はじめに

こんにちは、キノコードです。
仕事をしていると、PDFの中に書かれているテキストを、別の書類に使いたいという場面はないでしょうか。
PDFファイルのごく一部を取り出すだけであれば、コピペでもできますが、選択範囲が多く、何ページにもわたるようなときはかなり面倒な作業です。
しかも画像をPDF化したファイルですと、文字や表などをコピペしようとしても、うまくできないことが多いかと思います。
そんなときに、テキストファイルに読み込むことができると、かなりの業務効率化につながるでしょう。
PDF形式でもらった文書の内容をテキスト化することで、携帯にメールで転送するなど、文書の一部をテキストで保存・転送・管理したりすることも可能になります。
この動画では、Pythonでpdfを読み取り、テキストファイルに書き起こす方法について説明します。
キノコードでは、この動画の他にも仕事の自動化の動画、株のデータ分析の講座も配信しています。
ご興味ある方はぜひそちらもご覧ください。
チャンネル登録がまだの方は、新着通知をもいきますのでチャンネル登録をお願いします。

PDF操作のライブラリについて

Pythonには、PDFのページ操作やテキストの抽出を行うためのいくつかのライブラリがあります。
代表例として、PyPDF2、pdfminer.six、Apache Tikaがあります。
それぞれのライブラリの特徴を見ていきましょう。
最も手軽に使用できるのは、PyPDF2 です。
PyPDF2 では、PDF のページの操作や、テキストの読み取りができます。
しかし、デメリットとして、日本語に対応していないため、日本語の PDF は読み取ることができません。
英数字の PDF の読み取りに適しています。
日本語の PDF を読み取るためには、PDFMiner か Apache Tika を使用することで、日本語のテキストを抽出することができます。
Apache Tika は、Java で開発されたドキュメント分析・抽出ツールです。
Tika は、エクセルや PDF など様々な形式のファイルからテキストを抽出できます。
ただし、Python で利用するには、 Java の実行環境もインストールする必要があります。
一方、PDFMiner は、ライブラリをインストールするだけで、日本語のテキストを抽出することができます。
ですので、今回は、PDFMiner ライブラリを使い、日本語の PDF を読み取り、テキストを抽出し、テキストファイルにする、ということをやってみましょう。

PDFMinerでテキストを抽出する

```
# !pip install pdfminer.six
```

```
# 必要なライブラリのインストール
from pdfminer.pdfpage import PDFPage
from pdfminer.pdfinterp import PDFResourceManager, PDFPageInterpreter
from pdfminer.converter import TextConverter
from pdfminer.layout import LAParams
```

まず、今回のプログラムで使用するライブラリをインストールしておきましょう。
ライブラリとは、よく使う機能や関数をまとめて簡単に使えるようにしたものです。
今回は、pdfminer というライブラリを使用します。
pdfminerは、PDF ファイルからテキストを抽出するためのライブラリです。
日本語のテキストも抽出できるのが特徴です。
このコードを実行すると、Jupyter lab でインストールできます。
実行します。
インストールできました。
次に、pdfminer ライブラリのモジュールからクラスを使用できるように、これらをインポートします。インポートした各クラスについては、後ほど説明します。
実行します。
インポートが完了しました。

```
# 読み込みたいPDFファイル(http://it-gyousei.com/download/doc/koyoukeiyaku.pdf)
input_file = open('sample.pdf', 'rb')
# 書き込み用のテキストファイル
output_file = open('output.txt', 'w')
```

次に、プログラムで使用するPDFを読み込みます。
今回は、サンプルとしてこの動画の書き起こし用のPDFファイルを使用します。 サンプルのPDFについては、書き起こしのWebサイトからダウンロードできます。概要欄に記載しますので、ダウンロード後にこのipynb ファイルと同じディレクトリに保存しましょう。
open関数の第一引数に、用意したPDFファイル「sample.pdf」を渡します。
第二引数にバイナリファイルを読み込むためのモードである「rb」を指定します。
そして変数input_fileに代入します。
バイナリファイルとは、テキストデータではなく、バイナリデータを持っているファイルのことです。
バイナリデータとは、コンピュータが理解できるように数字と記号に変換したものです。 rが読み込みで、wが書き込みです。bがバイナリファイルを意味します。

次に、書き込み用のテキストファイルを用意します。
open 関数の第一引数に、ファイル名を「output.txt」として指定します。
第二引数に、書き込みモードの「w」を指定し、変数 output_file に代入します。

```
# Layout Analysisのパラメーターを設定
laparams = LAParams()
laparams
```

次に、LAParamsというクラスでインスタンスを作成し、変数laparamsに代入します。
LAParams クラスを設定することで、例えば、テキストを 1 文字ずつではなく、1行や、複数行をひとかたまりとして値を取得したりすることができます。
きれいにテキストを抽出するためには、必ず設定するクラスだと抑えておきましょう。
表示してみましょう。実行します。 このように、テキストを抽出するためのルールが設定されました。例えば、文字と文字の間隔、単語と単語の間隔、行間隔などです。 今回はデフォルトの値に設定しましたが、文書に応じて細かく調整をすることができます。

```
# 共有リソースを格納するPDFリソースマネージャーオブジェクトを作成
resource_manager = PDFResourceManager()
resource_manager
```

次に、PDFResourceManager クラスから resource_manager というインスタンスを作成します。
PDFResourceManager クラスは、フォントや画像などの共有リソースを格納するために使用されます。
PDFResourceManager クラスを使用することで、PDF ファイルのテキストのフォントや画像などの情報を、テキストファイルと共有することができます。 このクラスも必ず設定するクラスだと、抑えておきましょう。

```
# テキストに変換
device = TextConverter(resource_manager, output_file, laparams=laparams)
device
```

次に、TextConverter クラスを使用します。
TextConverter は、テキストに変換するためのクラスです。
第一引数には、PDFResourceManager クラスのオブジェクトのresource_managerを、第二引数には出力用のファイルoutput_fileを、第三引数の laparams には、LAParams()のオブジェクトのlaparamsを指定します。
これで生成したインスタンスを変数deviceに代入します。
オブジェクトが生成されました。

```
# ページの内容を処理するためのPDFインタプリタオブジェクトを作成
interpreter = PDFPageInterpreter(resource_manager, device)
interpreter
```

次に、PDFPageInterpreter（PDFページインタープリター） クラスを使用します。
PDFPageInterpreter は、テキストに変換したものをコンピュータが読み取れるように変換するためのクラスです。
第一引数には、PDFResourceManager クラスのオブジェクトのresource_managerを、第二引数には TextConverter のオブジェクトのdeviceを指定します。
ここで生成したインスタンスを変数interpreterに代入します。
これで、テキストが解析可能な状態のオブジェクトになりました。

```

```

```
# ドキュメントに含まれる各ページを処理
for page in PDFPage.get_pages(input_file):
    interpreter.process_page(page)
```

では最後に、1ページずつテキストを抽出して出力をしましょう。 forで、1ページごとに順に処理をします。
次に、PDFPage クラスと、get_pages メソッドを使用します。
PDFPage.get_pages()にファイルオブジェクトを指定して、PDFPage オブジェクトを順に取得することができます。
成功していると、同じディレクトリに「output.txt」というテキストファイルが作成されているはずです。
実行します。
テキストファイルを開いて確認してみましょう。
PDFファイルのテキストが抽出され、テキストファイルに書き出すことができました。 文字化けなどもなさそうです。これで完了です。

おわりに

レッスンは以上です。
いかがでしたでしょうか？手動でやると大変なことも、このようにプログラムで解決できることはたくさんあります。
まずは身近な文書から試してみて、どんどんプログラムを改善していくと、どんなファイルにも対応ができるようになります。
ぜひ業務を効率化して、ご自身にしかできないことをやるということにつなげてみてください。
キノコードでは、今後このような動画を配信予定です。
チャンネル登録がまだの方は、ぜひチャンネル登録をお願いいたします。 それではまた次のレッスンでお会いしましょう。